REMARKS

Applicant respectfully requests reconsideration of this application. Claims 1-27 are pending.

Claims 1-4, 6, 8-15, 17, 20-22, and 27 have been amended. Claims 5, 7, 16, 18, and 23-26 have been cancelled. No claims have been added.

Therefore, claims 1-4, 6, 8-15, 17, 19-22, and 27 are now presented for examination.

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al. and Kroun, et al.

The Examiner rejected claims 1, 3, 4, 7-10, 12-14, 16-18, and 22-27 under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,988,140 of Chintalapati et al. ("*Chintalapati*") in view of U.S Patent No. 6,711,643 of Park et al. ("*Park*") in further view of U.S Patent No. 6,584,560 of Kroun, et al. ("*Kroun*"). (referred to as "Jack" in the Final Office Action.) Claims 7, 16, 18, and 23-26 have been cancelled.

Claim 1, as amended here, is as follows:

1. A method comprising:

initializing a computer system, the computer system including an application processor to execute applications and processes in the computer system and a bootstrap processor to initialize the computer system;

declaring the application processor to be dedicated to handling a polling function for a timer interrupt process for the computer system, wherein a normal execution thread is to be processed by the bootstrap processor, and wherein the timer interrupt process is the only method of asynchronous event handling that is available to the computer system;

setting a timer for a plurality of time intervals for the timer interrupt process;

calling a polling function at the end of each of the plurality of time intervals, the polling function being performed by the application processor, the polling function to determine if any special events have occurred; and

if the polling function results in a positive result, processing the results of the polling function with the bootstrap processor.

Thus, as amended, claim 1 provides for initializing a computer system, with the computer system including an application processor and a bootstrap processor. In this process, the application processor is dedicated to handling a polling function for a timer interrupt process for the computer system, with a normal execution thread to be processed by the bootstrap processor. The timer interrupt process is the only method of asynchronous event handling that is available to the computer system. A timer is set for a plurality of time intervals for the timer interrupt process, and a polling function is called at the end of each of the plurality of time intervals, the polling function being performed by the application processor, the polling function to determine if any special events have occurred. If the polling function results in a positive result, the results of the polling function are processed with the bootstrap processor.

It is respectfully submitted that the cited references do not teach or reasonably suggest these claim elements.

Advisory Action – The Advisory Action indicates disagreement with the arguments previously presented of the Applicant as follows:

(1) This relates to the combination of the references. The Advisory Action indicates that "Both Chintalapati and Kroun teach the system for handling or managing

the interrupt of communications between the processes." Applicant argues this is not correct because, as shown below, Applicant submits that *Chintalapati* does not relate to interrupts of any kind.

(2) This relates to the argument of the Applicant that *Kroun* does not relate to timer interrupts. The Advisory Action states that: "Kroun teaches program routines designed to facilitate the communications of the processors with the peripherals through interrupts." It is submitted by the Applicant that this is not sufficient. As shown in more detail below, *Kroun* does relate to interrupts, but not to timer interrupts. Any program routines presented by the reference do not provide timer interrupts. Further, claim 1 provides that the timer interrupt process is the only method of asynchronous event handling that is available to the computer system. None of references provide for a system that operates without other interrupt processes.

The cited references are the following:

(1) *Chintalapati* - The *Chintalapati* reference is being cited as showing elements of the claims including a first processor, a timer interrupt process, with the first processor to handle a polling function for a timer interrupt process, and setting a time for a plurality of time intervals.

However, it is again submitted that *Chintalapati* does not discuss the concept of timer interrupts, or interrupts of any kind. The term "interrupt" is not used anywhere in the reference. The reference regards the servicing of idle connections, not the handling of interrupts. It is submitted that the reference is not relevant to the technology addressed in claim 1. Claim 1 does not merely discuss polling operations, but rather polling operations for a timer interrupt process, which is not addressed in the reference.

The *Chintalapati* reference does discuss a processing resource, which is a logical entity used by the server to service connections (*Chintalapati*, col. 6, lines 45-49), and a poll manager that receives idle connections from worker threads and passes active connections to a work queue (*Chintalapati*, col. 6, lines 63-67). The problem that is addressed by the reference regards the fact that there are times when a connection is idle and thus does not require servicing by the processing resource. (*Chintalapati*, col. 4, lines 52-60) The reference is demonstrating a system and process for disassociating the idle connections from the processing resources, and monitoring the idle connections for activity. This can be seen from the process illustrated in Figures 3A and 3B, which include the determination that a connection is idle 310, and the passing of the idle connection to the poll manager 320 for association with a poll subset 330. The poll adapter polls the connections in the poll subset 350 and returns the poll subset and events to a poll thread. The poll thread determines whether the connection is still idle or active 370, and whether an event is a closed connection 380. If a connection is active and not closed, the poll thread passes the connection to a work queue 390 and a worker thread picks up the connection for servicing 394. As is apparent from this discussion, the elements discussed are generally process elements.

The Final Office Action indicates that *Chintalapati* "does not explicitly teach that the [] second process as the second processor." It is submitted that this is not a sufficient basis for rejection because it inappropriately equates a "process" with a "processor". The reference discusses a "poll manager", which appears to be the element that the Final Office Action is referring to as the "process" that is performing the polling function. The

Chintalapati reference does make references to a processor, but this is not with regard to the poll manager. The reference indicates the following regarding the poll manager:

> FIG. 2, there is also a poll manager 250. Poll manager 250 receives idle connections from worker threads 240, 242, 244 and passes active connections to work queue 220 where the active connections wait for servicing by worker threads 240, 242, 244. Poll manager 250 includes a plurality of poll subsets, of which only poll subsets 260, 262 are shown for the sake of simplicity. Poll subsets 260, 262 hold the idle connections that are passed from worker threads 240, 242, 244 to poll manager 250.
>
> Because idle connections may be held in poll subsets and active connections may be serviced by worker threads, the total number of connections managed by server process 210 may be more than the number of worker threads that are available for server process 210. For example, in a typical system using the synchronous I/O model that dedicates a worker thread to each connection, the number of connections that can be serviced by server process 210 is limited by the number of work threads. However, in a system that incorporates poll manager 250, the capability of passing off idle connections to poll manager 250 can significantly increase the number of connections that can be serviced by server process 210. As a result of using poll manager 250 to track idle connections, server process 210 may be described as multiplexing the connections because more connections may be handled than there are processing resources to service the connections.

(*Chintalapati*, col. 6, line 63 to col. 7, line 4) (emphasis added) Further, "Server process 210 includes a poll adapter 280 that is communicatively coupled to poll manager 250." (*Chintalapati*, col. 7, lines 31-32) As to the question of what the poll manager actually is:

> According to one embodiment, connections are passed off from processing resources to a poll manager. The poll manager is a logical entity that may be implemented in software, hardware, or a combination thereof. The poll manager is responsible for monitoring idle connections to determine if and when the connections become active again. The poll manager includes poll threads that may perform the monitoring function and which may be responsible for timing out and closing connections, as discussed below. The poll threads may close connections in response to receiving a closed connection indication from a client.

(*Chintalapati*, col. 11, lines 11-21) Thus, the poll manager is a logical entity that may include hardware as well as software. The poll manager is never referred to as a separate processor, and is never suggested to act as a separate processor. The polling manager has only limited functions, and is described not as a processor, but simply as a part of a process. For example, Figure 2 of *Chintalapati* illustrates the poll manager 250 as a part of the server process 210.

It is submitted that the distinction between a "process" and a "processor" is very significant. A "process" may be run by a "processor", but the meaning of the terms is very different. By the reasoning of the Final Office Action, any time a "process" is discussed this may be used to imply that a separate processor exists to perform the process, which is not reasonable. The reference does not support this conclusion.

Thus, the combination suggested by the Final Office Action requires that the connection process that is described in *Chintalapati* be replaced with a processor for an interrupt, and there is no logical relation between these elements. As indicated above, the polling functions discussed in the reference do not relate to a timer interrupt, but rather to different technology regarding disassociating connections from processing resources.

(2) *Park* - According to the Final Office Action, the *Park* reference is provided because the reference teaches a second processor. Specifically, the reference cites to the slave processor 100 described in the reference, with the Final Office Action citing the following:

> The master processor 300 reads the status register of the vectored interrupt controller 400 and analyzes the source of the interrupt so as to process the interrupt received through the IRQ signal. If the interrupt is requested by a functioning unit controlled by the first ARM processor 100 designated as the slave processor according to the analyzed results, the master processor 300 sends both an interrupt redirection request and content of corresponding interrupt to the interrupt redirection unit 200 using the internal bus 900.

(*Park*, col. 5, lines 42-49) The *Park* reference regards an interrupt redirection apparatus and method for inter-processor communication. The apparatus described includes a plurality of ARM processors.

However, what is occurring in *Park* is that the master processor is redirecting interrupts, including the redirection of interrupts to the slave processor. This has no relation to the designation of an application processor to handle a polling function for a timer interrupt process for the computer system. The master processor initially handles interrupts, but redirects the interrupts to a first processor if the interrupt is requested by a functioning unit that is controlled by the first processor.

Thus, the *Park* reference regards a different kind of interrupt system, which, rather than having a processor that is designated for a polling function of a time interrupt processing, describes a processor (the master) that may receive multiple interrupts, but redirects the interrupts to other processors. *Park* is not relevant to a timer interrupt.

(3) *Kroun* (referred to as *Jack* in the Final Office Action) – The Final Office Action indicates that *Chintalapati* and *Park* do not teach initializing a computer system including a first processor and a second processor, and the Final Office Action cites to the *Kroun* reference for this element.

Kroun regards a method and system for booting a multiprocessor computer. The described system includes a memory bus used by processors to communicate with a main memory, and a second bus 30 (the APIC bus) that connects the processors to an interrupt controller 34 (the IO APIC module). The operation of the system is intended to provide for a process for booting the processors:

> A computer system is provided that can boot a multiprocessor system without reference to a hardwired precedence among the processors. The computer system includes a plurality of computer processors. A memory bus allows the processors to communicate with a main memory. A second bus connects the processors to an interrupt controller. The second bus includes at least bus request lines. An initialization control circuit that can read and assert signals on the bus request lines is provided. …

(Kroun, col. 2, lines 54-62) In this system, there is also an interrupt arrangement that provides for delivery of interrupts to each of the processors: "The APIC bus 30 is also connected to an IO APIC module 34 that is connected to the second bus bridge 26. During operation the computer system 10 receives interrupt requests from peripheral devices through the IO APIC Module 34. The processors 12 receive the interrupts from the APIC bus 30. More specifically, the processors 12 receive interrupt packets from the APIC bus 30. Those packets are cracked by the local APICs 40 (see FIG. 3)." (*Kroun*, col. 4, lines 20-28) Thus, in this described system an interrupt process exists, with a bus

that is used to provide interrupts and an interrupt controller that is used to deliver the interrupt requests to the processors. Further, there may be multiple local interrupt controllers: "A more specific computer system is also provided in which each processor includes a local interrupt controller. The local interrupt controllers are connected to the bus request lines." *Kroun*, col. 3, lines 4-7)

However, again the *Kroun* reference has no relation to timer interrupts, nor is there any reason to combine this system with a timer interrupt when a different type of interrupt system already exists. Rather, the interrupts are being handled in a completely different fashion, with interrupts being sent to controller, which then delivers the interrupts to the appropriate processors. With this interrupt system, it is not apparent what function a timer interrupt would provide – the reference provides for a specific system for delivery of interrupts, including an interrupt controller to receive and distribute the receiver interrupts.

In summary, the cited references provide the following:

(1) *Chintalapati* – Provides a system for servicing connections by disassociating processing resources from idle connections, and monitoring the idle connections for activity. Monitors the idle connections using a poll adapter process. Does not discuss interrupts.

(2) *Park* – Regards an interrupt redirection apparatus and method for inter-processor communication. Provides for redirecting interrupts from a master processor to a slave processor. Does not discuss timer interrupts – utilizes a different kind of interrupt system.

(3) *Kroun* – Discusses a system that provides for booting of a multi-processor system, including an interrupt controller and bus used for interrupt messages. Does not discuss timer interrupts – utilizes a different kind of interrupt system.

Thus, it is submitted that none of the references address a timer interrupt system, and definitely not a computer system in which only a timer interrupt is available. The rejection is based on a first system that involves polling by a certain process, which is combined with an interrupt system that utilizes a processor to forward interrupts, and a system that provides for booting of a multi-processor system. It is again submitted that these elements do not teach or reasonably suggest the limitations of claim 1.

Rationale for Combination – It is again submitted that the rationales provided for the combination of references are not legally sufficient. The Final Office Action indicates "It would have been obvious to one of the ordinary skill in the art at the time the invention was made to modify the teaching of Chintalapati with Park to incorporate the feature of second processor because this utilizes the inter-processor communication exclusive bus for connecting between exclusive controllers."

It is respectfully submitted that what is described is simply an element of the *Park* reference, and there is no rational basis for combining this with the *Chintalapati* reference. There is no indication why this feature would make any sense with regard to *Chintalapati*, which does not discuss interrupts. What is discussed in *Chintalapati* is a system polling of connections, and the inter-processor bus to connect controllers of the processors is not relevant to this concept.

The Final Office Action further provides that: "It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the teaching of

Chintalapati and Park with [Kroun] to incorporate the feature of initializing a computer system including a [first] processor and a second processor because this achieves greater efficiency by assigning each component of such a task to a different processor so that they can be performed in parallel." This is not a sufficient argument because it is not relevant to any teachings of *Chintalapati* or *Park*. If this were a sufficient basis for combination, then under the same reasoning *Kroun* could be combined with any reference that includes multiple computer tasks – the reason providing for combining is simply that the result is more efficient because components of tasks are divided among processors. This does not provide a legally sufficient basis to combine *Kroun* with *Chintalapati* or *Park*.

Claims 12 and 13 – For completeness, it is again noted that the Final Office Action rejects claims 12 and 13 on the basis of "Sugahara". This was a reference addressed in the previous Office Action, and it was Applicant's understanding the that Final Office Action is not based on this reference as it is not identified in any other portion of the Final Office Action. However, these citations appear to refer to *Sugahara*. If these claims are rejected, the Applicant respectfully requests clarification of the basis for the rejection.

It is thus submitted that claim 1 is patentable over *Chintalapati* in view of *Park* and further in view of *Kroun*. The arguments presented above also apply to independent claims 8, 14, and 22, and such claims are also allowable.

The remaining rejected claims are dependent claims, which, in addition to other differences with the cited references, are allowable as being dependent on the allowable base claims.

The Applicant hereby again addresses the arguments presented for the other claim rejections:

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al., Kroun, et al. and Nguyen

The Examiner rejected claims 2 and 11 under 35 U.S.C. 103(a) as being unpatentable over *Chintalapati* in view of *Park* in further view of *Kroun* and in further view of U.S. Patent No. 7,296,069 of Nguyen ("*Nguyen*"). (referred to in the Final Office Action "Hoa".)

The rejected claims are dependent claims, and, while having other differences with the cited references, are allowable as being dependent on the allowable base claims.

Chintalapati, *Park*, and *Kroun* have been discussed above. While this reference is cited for other reasons, it is again submitted that *Nguyen* does not contain any teaching or suggestion of the elements shown to be missing from the other references. *Nguyen* regards a method and system for monitoring faults in network interface cards. Specifically, the *Nguyen* reference regards initializing data structures for tracking the status of one or more network interface cards to be monitored; initiating monitoring of the one or more network interface cards; ascertaining a configurable polling interval; determining if a shutdown condition has occurred; monitoring the status of the one or more network interface cards when a shutdown condition has not occurred; and clearing all resources when a shutdown condition has occurred. The reference contains no discussion of timer interrupts, or interrupts of any kind.

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al., Kroun, et al. and Karnik et al.

The Examiner rejected claims 5 and 15 under 35 U.S.C. 103(a) as being unpatentable over *Chintalapati* in view of *Park* in further view of *Kroun* and in further view of U.S. Patent No. 5,724,527 of Karnik et al. ("*Karnik*"). Claim 5 has been cancelled.

The rejected claims are dependent claims, and, while having other differences with the cited references, are allowable as being dependent on the allowable base claims.

Chintalapati, *Park*, and *Kroun* have been discussed above. While this reference is cited for other reasons, it is again submitted that *Karnik* does not contain any teaching or suggestion of the elements shown to be missing from the other references. *Karnik* regards a fault-tolerant boot strap mechanism for a multiprocessor system, and specifically utilizes message passing between microprocessors to dynamically determine a system-generated bootstrap processor, and discussed the utilization of a local interrupt controller unit of each processor to dynamically determine the bootstrap processor. Thus, the reference regards the selection of the bootstrap processor, and does discuss interrupt elements in this process, but does not contain any discussion of timer interrupts.

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al., Kroun, and Yamamoto

The Examiner rejected claim 6 under 35 U.S.C. 103(a) as being unpatentable over *Chintalapati* in view of *Park* in further view of *Kroun* and in further view of Japanese Patent No. JP405252374A of Yamamoto et al. ("*Yamamoto*").

The rejected claim is a dependent claim, and, while having other differences with the cited references, is allowable as being dependent on the allowable base claim.

Chintalapati, *Park*, and *Kroun* have been discussed above. While this reference is cited for other reasons, it is again submitted that *Yamamoto* does not contain any teaching or suggestion of the elements shown to be missing from the other references. While it is difficult to fully discern what is contained in this reference, *Yamamoto* generally regards "facsimile equipment", and appears to regard "executing the input processing of the succeeding mutipolling start date and time and the group number in parallel with polling reception processing." However, it is unclear what precisely this means. It appears that this regards processing of a start date and time in parallel with polling reception processing, but this does not appear to have any relation to interrupt processing. It is not clear what is being polled or what processing is being conducted.

It is submitted that it is not clear how the *Yamamoto* reference applies here, and thus the reference does not provide a sufficient basis for rejection of the claims.

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al., Kroun, and Yang et al.

The Examiner rejected claims 19 under 35 U.S.C. 103(a) as being unpatentable over *Chintalapati* in view of *Park* in further view of *Kroun* and in further view of U.S. Patent No. 7,003,610 of Yang et al. ("*Yang*").

The rejected claim is a dependent claim, and, while having other differences with the cited references, is allowable as being dependent on the allowable base claim.

Chintalapati, *Park*, and *Kroun* have been discussed above. While this reference is cited for other reasons, it is again submitted that *Yang* does not contain any teaching or

suggestion of the elements shown to be missing from the other references. *Yang* regards handling shared resource writes that are arriving via non-maskable interrupts. Specifically, the reference regards handling write requests in a manner that services the write request and does not disturb the integrity of shared resources or mask the interrupt request. There does not appear to be any discussion in the reference that regards the claim elements found to be missing from *Chintalapati*, *Park*, and *Kroun.*

Claim Rejection under 35 U.S.C. §103

Chintalapati et al in view of Park et al., Kroun, et al. and Hokenek et al.

The Examiner rejected claims 20 and 21 under 35 U.S.C. 103(a) as being unpatentable over *Chintalapati* in view of *Park* in further view of *Kroun* and in further view of U.S. Patent No. 6,971,103 of Hokenek et al. ("*Hokenek*").

The rejected claims are dependent claim, and, while having other differences with the cited references, are allowable as being dependent on the allowable base claim.

Chintalapati, *Park*, and *Kroun* have been discussed above. It is again submitted that *Hokenek* does not contain any teaching or suggestion of the elements shown to be missing from the other references. The *Hokenek* reference regards inter-thread communications using a shared interrupt register, and discusses a multithreaded processor that includes an interrupt controller for processing a cross-thread interrupt directed from a requesting thread to a destination thread. It does not appear that the reference has any relation to timer interrupts.

Conclusion

Applicant respectfully submits that the rejections have been overcome by the amendment and remark, and that the claims as amended are now in condition for

allowance. Accordingly, Applicant respectfully requests the rejections be withdrawn and the claims as amended be allowed.

Invitation for a Telephone Interview

The Examiner is requested to call the undersigned at (503) 439-8778 if there remains any issue with allowance of the case.

Request for an Extension of Time if Needed

The Applicant respectfully petitions for an extension of time to respond to the outstanding Final Office Action pursuant to 37 C.F.R. § 1.136(a) should one be needed. Please charge any fee to our Deposit Account No. 02-2666.

Charge our Deposit Account

Please charge any shortage to our Deposit Account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: April 10, 2008

/Mark C. Van Ness/
Mark C. Van Ness
Reg. No. 39,865

1279 Oakmead Parkway
Sunnyvale, CA 94085-4040

(503) 439-8778